Soft Comput (2015) 19:1215–1227
DOI 10.1007/s00500-014-1336-3

METHODOLOGIES AND APPLICATION

A mutative-scale pseudo-parallel chaos optimization algorithm

Xiaofang Yuan · Xiangshan Dai · Lianghong Wu

Published online: 21 June 2014

Abstract Chaos optimization algorithms (COAs) utilize the chaotic map to generate the pseudo-random sequences mapped as the decision variables for global optimization applications. Many existing applications show that COAs escape from the local minima more easily than classical stochastic optimization algorithms. However, the search efficiency of COAs crucially depends on appropriately starting values. In view of the limitation of COAs, a novel mutative-scale pseudo-parallel chaos optimization algorithm (MPCOA) with cross and merging operation is proposed in this paper. Both cross and merging operation can exchange information within population and produce new potential solutions, which are different from those generated by chaotic sequences. In addition, mutative-scale search space is used for elaborate search by continually reducing the search space. Consequently, a good balance between exploration and exploitation can be achieved in the MPCOA. The impacts of different chaotic maps and parallel numbers on the MPCOA are also discussed. Benchmark functions and parameter identification problem are used to test the performance of the MPCOA. Simulation results, compared with other algorithms, show that the MPCOA has good global search capability.

Keywords Chaotic map · Chaos optimization algorithm (COA) · Parallel chaos optimization algorithm (PCOA) · Cross operation · Merging operation

1 Introduction

From the mathematical aspect, chaos is defined as a pseudo-random behavior generated by nonlinear deterministic systems (Yang et al. 2006). Generally speaking, chaos has several important dynamical characteristics, namely, the sensitive dependence on initial conditions, pseudo-randomness, ergodicity, and strange attractor with self-similar fractal pattern (Yang et al. 2006, 2007, 2012; Li and Jiang 1998; Yuan and Wang 2008). Recently, chaos theory has been used to develop novel global optimization techniques, and particularly, in the specification of chaos optimization algorithms (COAs) (Li and Jiang 1998; Yang et al. 2007, 2012, 2014; Yuan and Wang 2008; Zhu et al. 2012; Hamaizia et al. 2012; Okamoto and Hirata 2013) based on the use of numerical sequences generated by means of chaotic map. Due to the dynamic properties of chaotic sequences, a lot of existing application results have demonstrated that COAs can carry out overall searches at higher speeds than stochastic ergodic searches that depend on the probabilities (Yang et al. 2012, 2014; Zhu et al. 2012; Hamaizia et al. 2012; Okamoto and Hirata 2013). Furthermore, COAs generally exhibit better numerical performance and benefits than stochastic algorithms. The main advantages of COAs include: (a) COAs escape from local minima more easily than classical stochastic optimization algorithms such as genetic algorithm (GA), simulated annealing (SA) and some meta-heuristics

Communicated by V. Loia.

This work was supported in part by the National Natural Science Foundation of China (No. 61104088, No. 61203309) and Young Teachers Promotion Program of Hunan University .

X. Yuan (✉) · X. Dai
College of Electrical and Information Engineering,
Hunan University, Changsha 410082, People's Republic of China
e-mail: yuanxiaofang126@126.com

L. Wu
College of Information and Electrical Engineering,
Hunan University of Science and Technology, Changsha 411201,
People's Republic of China

algorithms including particle swarm optimization (PSO), ant colony optimization algorithm (ACO), differential evolution (DE), and so on (Yang et al. 2014); (b) COAs don't depend on the strict mathematical properties of the optimization problem, such as continuity, differentiability; (c) COAs are easy to be implemented and the execution time of COAs is short (Li and Jiang 1998; Yang et al. 2007, 2012; Yuan and Wang 2008).

In addition to the development of COAs, chaos has also been integrated with meta-heuristic algorithms, such as: chaotic harmony search algorithm (Alatas 2010; Askarzadeh 2013), chaotic ant swarm optimization (Liu et al. 2013; Wan et al. 2012; Li et al. 2012), chaotic particle swarm optimization (Wu 2011; Cheng et al. 2012; Pluhacek et al. 2014), chaotic evolutionary algorithm (Ho and Yang 2012; Arunkumar and Jothiprakash 2013), chaotic genetic algorithms (Kromer et al. 2014; Ma 2012), chaos embedded discrete self organizing migrating algorithm (Davendra et al. 2014), chaotic differential evolution (Coelho and Pessoa 2011), chaotic firefly algorithm (Coelho and Mariani 2012), chaotic simulated annealing (Chen 2011; Hong 2011), chaos-based immune algorithm (Chen et al. 2011). The simulation results and applications have also shown the high efficiency, solutions diversity and global search capability of chaos-based optimization algorithms.

An essential feature of the chaotic sequence is that small change in the parameter or the starting value leads to the vastly different future behavior. Since the chaotic motions are pseudo-random and chaotic sequences are sensitive to the initial conditions, therefore, COAs' search and converge speed are usually effected by the starting values. In view of the limitation of COAs, a kind of parallel chaos optimization algorithm (PCOA) has been proposed in our former studies (Yuan et al. 2007, 2012, 2014), and simulation results show PCOA's superiority over original COAs. The salient feature of PCOA lies in its pseudo-parallel mechanism. In the PCOA, multiple stochastic chaos variables are simultaneously mapped onto one decision variable, so PCOA searches from diverse initial points and detracts the sensitivity of initial conditions. Although the PCOA in (Yuan et al. 2007, 2012, 2014) can easily escape from local minima, its precise exploiting capability is insufficient. Therefore, PCOA is combined with local search method, like simplex search method (SSM) and harmony search algorithm (HSA). However, this kind of hybrid PCOA with local search method is far from perfect as: the local search method has slow efficiency, the proper switch point from PCOA to local search method usually affects the search performance. In addition, parallel variables in PCOA search independently without information exchange.

To improve original PCOA, a mutative-scale pseudo-parallel chaos optimization algorithm (MPCOA) with cross and merging operation is proposed in this paper. Both cross and merging operation can exchange information within population and produce new potential parallel variables, which are different from those generated by chaotic sequences. In addition, mutative-scale search space is used to continually reduce the search space. Using cross and merging operation as well as mutative-scale strategy, MPCOA achieves a good balance between exploration and exploitation, without hybrid with local search method. The impacts of different chaotic maps and parallel numbers on the MPCOA are also discussed.

The rest of this paper is organized as follows. Section 2 briefly describes chaotic maps. The PCOA approach is introduced in Sect. 3. Section 4 gives presentation of the proposed MPCOA. The MPCOA is tested with benchmark functions and parameter identification problem in Sects. 5 and 6, respectively. Conclusions are presented in Sect. 7.

2 Chaotic map

One-dimensional maps are the simplest systems with the capability of generating chaotic behaviors. Eight well-known one-dimensional maps which yield chaotic behaviors are introduced as follows (Yuan et al. 2014; He et al. 2001; Tavazoei and Haeri 2007).

1. Logistic map: Logistic map generates chaotic sequences in (0,1). This map is also frequently used in the COAs and it is given by:

$$x_{n+1} = 4x_n(1 - x_n) \tag{1}$$

2. Tent map: Tent chaotic map is very similar to Logistic map, which displays specific chaotic effects. Tent map generates chaotic sequences in (0,1) and it is given by:

$$x_{n+1} = \begin{cases} x_n/0.7, & x_n < 0.7 \\ (\frac{10}{3})x_n(1 - x_n), & x_n \geq 0.7 \end{cases} \tag{2}$$

3. Chebyshev map: Chebyshev chaotic map is a common symmetrical region map. It is generally used in neural networks, digital communication and security problems. Chebyshev map generates chaotic sequences in [−1,1]. This map is formally given by:

$$x_{n+1} = \cos(5\cos^{-1} x_n) \tag{3}$$

4. Circle map: Circle chaotic map was proposed by Kolmogorov. This map describes a simplified model of the phase locked loop in electronics (Tavazoei and Haeri 2007). This map is formally given by:

$$x_{n+1} = x_n + 2.5 - \left(\frac{5}{2\pi}\right)\sin(2\pi x_n) \mod (1) \tag{4}$$

Circle map generates chaotic sequences in (0,1). In Eq. (4), x_{n+1} is computed mod 1.

5. Cubic map: Cubic map is one of the most commonly used maps in generating chaotic sequences in various applications like cryptography. Cubic map generates chaotic sequences in (0,1) and it is formally given by:

$$x_{n+1} = 2.59x_n\left(1 - x_n^2\right) \tag{5}$$

6. Gauss map: Gauss map is also one of the very well known and commonly employed map in generating chaotic sequences in various applications like testing and image encryption. Gauss map generates chaotic sequences in (0,1), and it is formally given by:

$$x_{n+1} = \begin{cases} 0, & x_n = 0 \\ \frac{1}{x_n} \bmod (1), & x_n \neq 0 \end{cases} \tag{6}$$

7. ICMIC map: Iterative chaotic map with infinite collapses (ICMIC) generates chaotic sequences in (−1,1) and it is formally given by:

$$x_{n+1} = \sin\left(\frac{70}{x_n}\right) \tag{7}$$

8. Sinusodial map: Sinusodial map generates chaotic sequences in (0,1) and it is given by:

$$x_{n+1} = \sin(\pi x_n) \tag{8}$$

The chaotic motions of these eight chaotic maps in two-dimension space (x_1, x_2) with 200 iterations are illustrated in Fig. 1. Here, the initial values of two chaos variables are: $x_1 = 0.152$, $x_2 = 0.843$. From Fig. 1, it can also be observed that the distribution or ergodic property of different chaotic maps is different. Therefore, the search performance of different chaotic maps differs from each other in view of convergence rate and accuracy (Yuan et al. 2012, 2014; He et al. 2001; Tavazoei and Haeri 2007).

3 PCOA approach

The salient feature of PCOA lies in its pseudo-parallel mechanism. In the PCOA, multiple stochastic chaos variables (like population) are simultaneously mapped onto one decision variable, and the search result is the best value of parallel multiple chaos variables.

Consider an optimization problem for nonlinear multi-modal function with boundary constraints as:

$$\min \ f(X) = f(x_1, x_2, \ldots, x_n), \quad x_i \in [L_i, U_i]. \tag{9}$$

where f is the objective function, and $X = (x_1, x_2, \ldots, x_n) \in R^n$ is a vector in the n-dimensional decision (variable) space (solution space), and the feasible solution space is $x_i \in [L_i, U_i]$, where L_i and U_i represent the lower and upper bound of the ith variable, respectively. PCOA evolves a stochastic population of N candidate individuals (solutions) with n-dimensional parameter vectors. The N is the population size, meanwhile, the number of parallel candidate individuals. In the following, the subscripts i and j stand for the decision variable and parallel candidate individual, respectively.

The process of PCOA based on twice carrier wave mechanism is described as follows. The first is the raw search in different chaotic traces, while the second is refined search to enhance the search precision.

3.1 PCOA search using the first carrier wave

Step 1: Specify the maximum number of iterations S_1 in the first carrier wave, random initial value of chaotic map $0 < \gamma_{ij}^{(0)} < 1$, initial iterations number $l = 0$, parallel optimum $P_j^* = \infty$ and global optimum $P^* = \infty$.

Step 2: Map chaotic map $\gamma_{ij}^{(l)}$ onto the variance range of decision variable by the equation:

$$x_{ij}^{(l)} = L_i + \gamma_{ij}^{(l)}(U_i - L_i) \tag{10}$$

Step 3: Compute the objective function value of $x_{ij}^{(l)}$, then update the search result. If $f(x_j^{(l)}) < P_j^*$, then parallel solution $x_j^* = x_j^{(l)}$, parallel optimum $P_j^* = f(x_j^{(l)})$. If $P_j^* < P^*$, then global optimum $P^* = P_j^*$, global solution $X^* = x_j^*$. This means that the search result is the best value of parallel candidate individuals.

Step 4: Generate next value of chaotic map using one of the chaotic maps in Eqs. (1), (2), (3), (4), (5), (6), (7), (8):

$$\gamma_{ij}^{(l+1)} = M(\gamma_{ij}^{(l)}) \tag{11}$$

Step 5: If $l \geq S_1$, stop the first carrier wave; otherwise $l \longleftarrow l + 1$, go to Step 2.

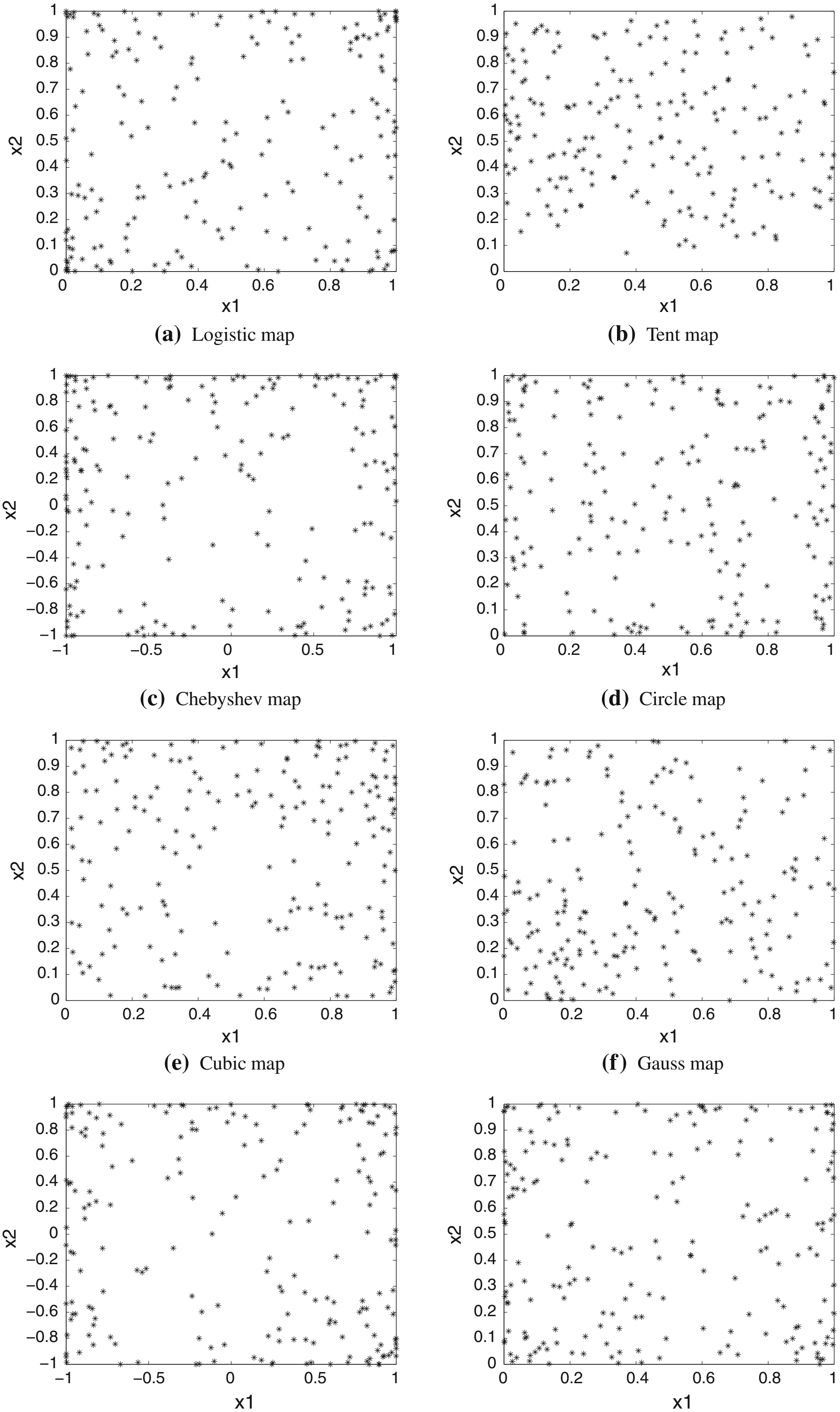
3.2 PCOA search using the second carrier wave

Step 1: Specify the maximum number of iterations S_2 in the second carrier wave, initial iterations number $l' = 0$, random initial value of chaotic map $0 < \gamma_{ij}^{(l')} < 1$.

Step 2: Compute the second carrier wave by the equation:

$$x_{ij}^{(l')} = x_j^* + \lambda_i(\gamma_{ij}^{(l')} - 0.5) \tag{12}$$

Step 3: Compute the objective function value of $x_{ij}^{(l')}$, then update the search result. If $f(x_j^{(l')}) < P_j^*$, then parallel



(a) Logistic map **(b)** Tent map

(c) Chebyshev map **(d)** Circle map

(e) Cubic map **(f)** Gauss map

(g) ICMIC map **(h)** Sinusodial map

Fig. 1 Chaotic maps in two-dimension space with 200 iterations solution $x_j^* = x_j^{(l')}$, parallel optimum $P_j^* = f(x_j^{(l')})$. If $P_j^* < P^*$, then global optimum $P^* = P_j^*$, global solution $X^* = x_j^*$.

Step 4: Generate next value of chaotic map $\gamma_{ij}^{(l'+1)}$ as in Eq. (11).

Step 5: If $l' \geq S_2$, stop the second carrier wave search process; otherwise $\lambda_i \longleftarrow t\lambda_i, l' \longleftarrow l' + 1$, go to Step 2.

The λ is an adjustable parameter and adjusts small ergodic range around x_j^*, and constant $t > 1$. It is difficult and heuristic to determine the appropriate value of λ_i, initial value of λ_i is usually set to $0.01(U_i - L_i)$ (Tavazoei and Haeri 2007).

4 MPCOA approach

A mutative-scale pseudo-parallel chaos optimization algorithm (MPCOA) with cross and merging operation is proposed in this section. Compared with original PCOAs, the proposed MPCOA has two obvious characteristics: (1) MPCOA doesn't need twice carrier wave search, only one carrier wave search can reach good solutions; (2) Cross operation, merging operation and mutative-scale search space are applied in MPCOA, without hybrid with local search method.

4.1 Cross operation within population

In the original PCOAs, all parallel variables search independently according to their respective chaotic sequences without information interaction. In the proposed MPCOA, the cross operation within population will be used to find new potential solutions. The cross operation is illustrated in Fig. 2. This kind of cross operation randomly chooses two parallel variables from parallel solutions x_j^* to exchange with each other, and it may produce new potential solutions which usually are different from those generated by chaotic sequences.

4.2 Merging operation within population

Even if original PCOAs have reached the neighborhood of global optimum, it needs to spend much computational effort to reach the optimum eventually by searching numerous points (Yuan and Wang 2008). The reason is that precise exploiting capability of PCOA is poor. For this reason, merging operation within population is employed in the MPCOA. The merging operation is illustrated in Fig. 3.

The merging operation within population from parallel solutions x_j^* can be denoted by:

$$x_{i1}^{(M)} = \gamma_{i1} * x_{i1}^* + (1 - \gamma_{i1}) * x_{i2}^* \tag{13}$$

where γ_{i1} is the chaotic map.

The merging operation randomly chooses two parallel variables from parallel solutions to merge, and it may produce new potential solutions for optimization problem. In essence, the merging operation is a kind of local exploiting search as shown in Eq. (13).

Both cross and merging operation within population are used as the supplement to the PCOA search. This means that if the new parallel variables after cross or merging operation have reached a better fitness than the original ones, the new

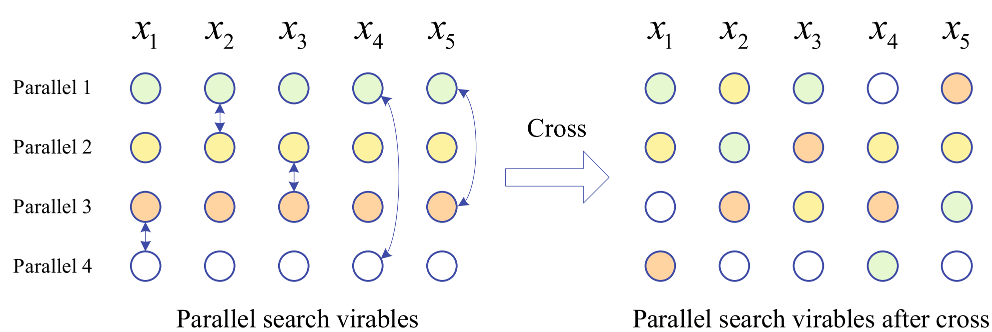


Fig. 2 Cross operation within population

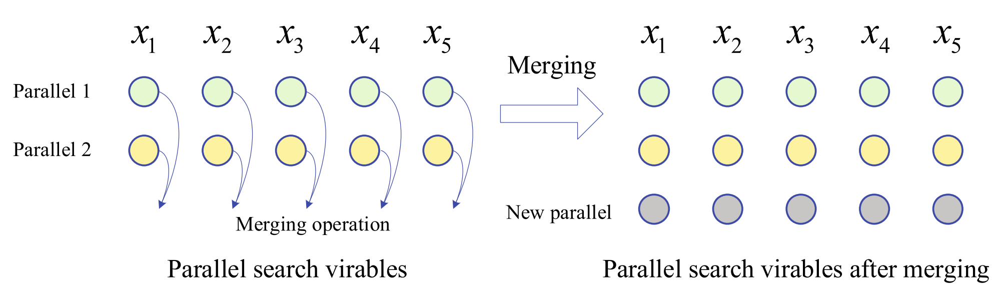


Fig. 3 Merging operation within population

parallel variables will replace the original ones. In another situation, if the new parallel variables after cross or merging operation bring a worse fitness than the original ones, the new parallel variables using cross or merging operation will be given up.

Both cross and merging operation within population are conducted at each iteration during the PCOA search procedure. Another problem is to choose how many parallel variables for cross or merging operation. The more cross or merging operation within population, the more diversity of parallel variables, while the more computing cost. In this paper, the cross operation rate is P_{cross} = 0.1–0.5, and the merging operation rate is P_{merging} = 0.1–0.5, i.e., about 10–50 % of parallel solutions have been executed the cross or merging operation.

4.3 Mutative-scale search

With the increase of iterations, the parallel solutions for optimization problem will be close to each other. So the precise exploiting search is the main task after certain iterations. Here, mutative-scale search using contractive search space is considered for precise exploiting search.

Mutative-scale search space is conducted by the following equations:

$$L_i' = x_i^* - \Phi(U_i - L_i) \tag{14}$$

$$U_i' = x_i^* + \Phi(U_i - L_i) \tag{15}$$

where Φ represents a mutative-scale factor which is a decreasing parameter with respect to iterations l described by:

$$\Phi = \left(\frac{l_{\max} - l}{l_{\max}}\right)^{\phi} \tag{16}$$

where ϕ is an integer usually set to 2–8.

To avoid L_i', U_i' exceeding bounds of search space $[L_i, U_i]$, the new search ranges are restricted to their bounds: If $L_i' < L_i$, then $L_i' = L_i$; if $U_i' > U_i$, then $U_i' = U_i$.

Then, the search space will be contracted for better accurate exploiting search, and the modified search space is used in the follow-up procedure as:

$$L_i = L_i' \tag{17}$$

$$U_i = U_i' \tag{18}$$

4.4 MPCOA implementation

The detailed implementation of the MPCOA is presented as follows. The process of MPCOA is also illustrated in Fig. 4.

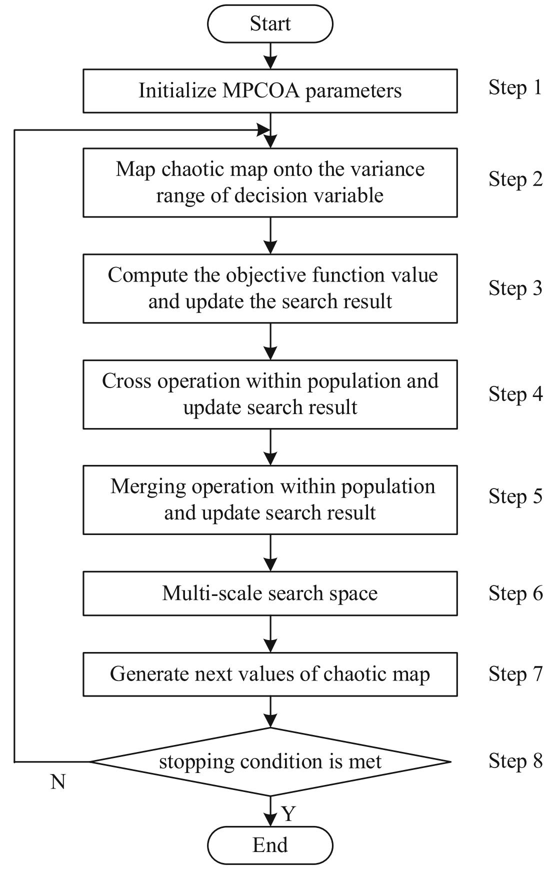


Fig. 4 Flowchart of the proposed MPCOA approach

Step 1: Specify the maximum number of iterations $l_{\max}$, random initial value of chaotic map $0 < \gamma_{ij}^{(0)} < 1$, set iterations number $l = 0$, parallel optimum $P_j^* = \infty$ and global optimum $P^* = \infty$.

Step 2: Map chaotic map $\gamma_{ij}^{(l)}$ onto the variance range of decision variable as in Eq. (10).

Step 3: Compute the objective function value of $X^{(l)}$, then update the search result. If $f(x_j^{(l)}) < P_j^*$, then parallel solution $x_j^* = x_j^{(l)}$, parallel optimum $P_j^* = f(x_j^{(l)})$. If $P_j^* < P^*$, then global optimum $P^* = P_j^*$, global solution $X^* = x_j^*$.

Step 4: Execute cross operation and produce new parallel variable $x_j^{(C)}$. Compute the objective function value of $x_j^{(C)}$, then update the search result. If $f(x_j^{(C)}) < P_j^*$, then $x_j^* = x_j^{(C)}$, $P_j^* = f(x_j^{(C)})$. If $P_j^* < P^*$, then $P^* = P_j^*$, $X^* = x_j^*$.

Step 5: Execute merging operation and produce new parallel variable $x_j^{(M)}$. Compute the objective function value of $x_j^{(M)}$, then update the search result. If $f(x_j^{(M)}) < P_j^*$, then $x_j^* = x_j^{(M)}$, $P_j^* = f(x_j^{(M)})$. If $P_j^* < P^*$, then $P^* = P_j^*$, $X^* = x_j^*$.

Step 6: Conduct mutative-scale search space $[L_i', U_i']$ as in Eqs. (14), (15), then update the search space $[L_i, U_i]$ as in Eqs. (17), (18).

Table 1 Benchmark functions

No.	Function	Function expression	Bounds	n	Optimum
1	Camel-back	$f_1(x, y) = (4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + (-4 + 4y^2)y^2$	$[-20, 20]$	2	-1.0316284
2	Schaffer	$f_2(x, y) = 0.5 - \frac{\sin^2\sqrt{x^2+y^2} - 0.5}{(1+0.001(x^2+y^2))^2}$	$[-20, 20]$	2	1.0
3	Goldstein	$f_3(x, y) = \{1 + (x + y + 1)^2(19 - 14x + 3x^2 - 14y + 6xy + 3y^2)\}$ $\times\{30 + (2x - 3y)^2(18 - 32x + 12x^2 + 48y - 36xy + 27y^2)\}$	$[-20, 20]$	2	3.0
4	Rastrigin	$f_4(x_i) = \sum_{i=1}^{n}\{x_i^2 - 10\cos(2\pi x_i) + 10\}$	$[-5, 5]$	3	0
5	Griewank	$f_5(x_i) = \frac{1}{4{,}000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-5, 5]$	20	0
6		$f_6(x_i) = \frac{1}{N}\sum_{i=1}^{n}(x_i^4 - 16x_i^2 - 5x_i)$	$[-10, 10]$	20	-78.3323314
7	Rosenbrock	$f_7(x_i) = \frac{1}{n}\sum_{i=1}^{n}\{10(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\}$	$[-10, 10]$	20	0
8	Ackley	$f_8(x_i) = -20\exp\left\{-0.2\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}\right\} - \exp\left\{\frac{\sum_{i=1}^{n}\cos(2\pi x_i)}{n}\right\} + 20 + e$	$[-10, 10]$	20	0

Step 7: Generate next value of chaotic map $\gamma_{ij}^{(l+1)}$ as in Eq. (11).

Step 8: If $l \geq l_{\max}$, stop the search process; otherwise $l \longleftarrow l + 1$, go to Step 2.

5 Benchmark functions simulation

5.1 Benchmark functions

Well-defined benchmark functions which are based on the mathematical functions can be used as objective functions to measure and test the performance of optimization algorithms. The efficiency and performance of the MPCOA are evaluated with eight common benchmark functions as in Table 1.

Among these benchmark functions, the former four have two or three decision variables, and the latter four have 20 decision variables. These functions are often multi-modal and several of them have many local minima, as illustrated in Fig. 5.

5.2 Different parallel numbers N for MPCOA

Like the population size to evolutionary computing, the parallel number N in MPCOA is one parameter that a researcher has to deal with. We still largely depend on the experience or trial-and-error approach to set parameters. Generally speaking, too small parallel number N leads to similar convergence of original COAs, while too large parallel number N is computationally costly. Now we will investigate how parallel number N will influence the MPCOA performance. The parameters in the MPCOA are: $P_{\text{cross}} = 0.5$, $P_{\text{merging}} = 0.5$, $\phi = 6$, different parallel numbers N have been investigated, tent map in Eq. (2) is used as the chaotic map. For different optimization problems, appropriate number of maximum iterations $l_{\max}$ of MPCOA is usually different. An appropriate $l_{\max}$ is usually chosen with multiple runs until finding the one which yields an appropriate result. In this simulation, $l_{\max}$ is chosen by trial with multiple runs and is labeled in Fig. 6. Convergence performance of MPCOA on benchmark functions with different parallel numbers ($N = 6,\ 12,\ 20,\ 30$) is shown in Fig. 6.

The simulation results on benchmark functions obtained by MPCOA with different parallel numbers N are reported in Table 2, which shows the statistical result in 20 runs. The 'Best', 'Worst' and 'Mean' represent the best, the worst and the average objective function value by MPCOA in 20 runs, respectively. The 'Rate' represents the success rate of reaching the global optimum X^* with the error less than 0.01.

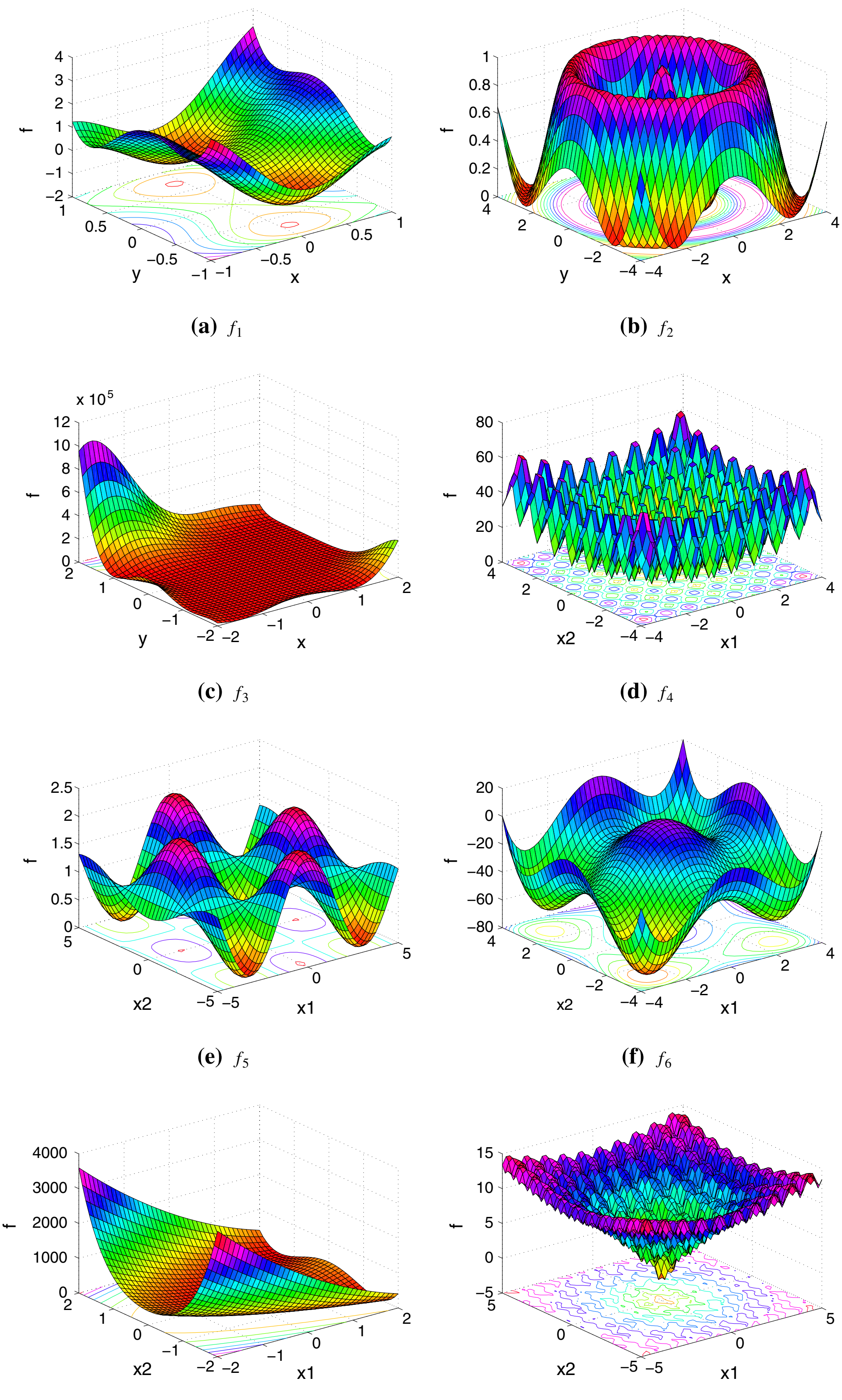
It can be seen from Fig. 6 and Table 2 that parallel number N has a direct impact on the search result of MPCOA. With the increase of parallel number N, the search speed of MPCOA is faster and the search results are better. For these benchmark functions, the proposed MPCOA achieves satisfactory success rate and global optimum when $N = 30$.

5.3 Different chaotic maps for MPCOA

The performance of MPCOA using different chaotic maps described in Eqs. (1), (2), (3), (4), (5), (6), (7), (8) will be investigated here. The MPCOA parameters are: $N = 20$, $P_{\text{cross}} = 0.5$, $P_{\text{merging}} = 0.5$, $\phi = 6$. The simulation results on benchmark functions by the MPCOA using different chaotic maps are reported in Table 3. It can be seen from Table 3 that there is very little difference with respect to different chaotic maps for MPCOA. From Table 3, it can be seen that the sinusodial map, tent map and Gauss map show better simulation results than other maps in these tests.

5.4 Comparison of MPCOA with other PCOAs

The proposed MPCOA is also compared with other PCOAs in Yuan et al. (2007, 2012). With the similar conditions and parameters as $P_{\text{cross}} = 0.5$, $P_{\text{merging}} = 0.5$, $\phi = 6$ and tent, map is used. The simulation results of success rate by



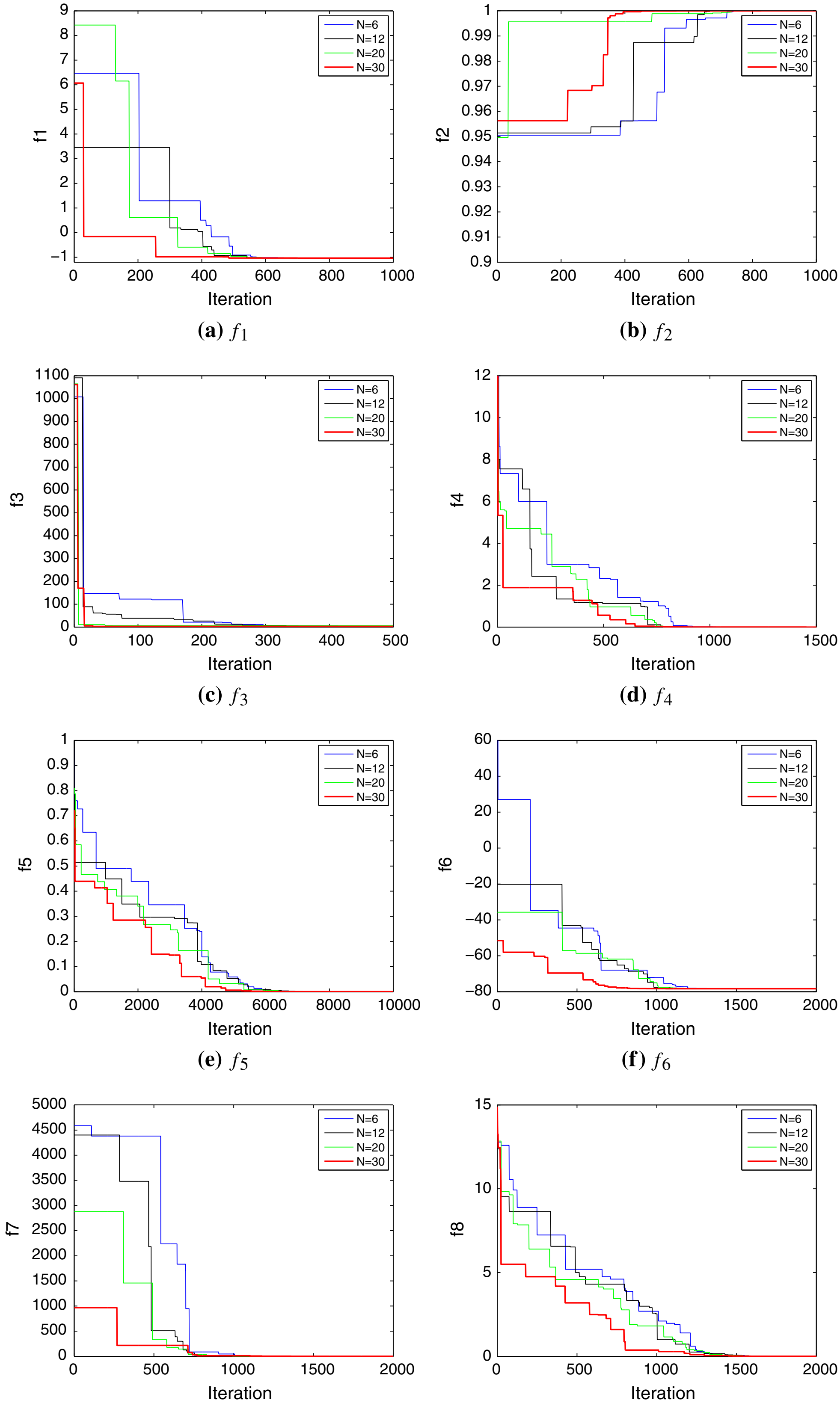
(a) f_1 **(b)** f_2

(c) f_3 **(d)** f_4

(e) f_5 **(f)** f_6

(g) f_7 **(h)** f_8

Fig. 5 Benchmark functions $f_1 - f_8$ illustrated in two-dimension space



(a) f_1 (b) f_2

(c) f_3 (d) f_4

(e) f_5 (f) f_6

(g) f_7 (h) f_8

Fig. 6 Convergence performance of MPCOA on benchmark functions with different parallel numbers N

Table 2 Simulation results obtained by MPCOA with different parallel numbers N

N	Stats	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
$N = 6$	Best	−1.031611	0.999924	3.000026	0.000023	0.000015	−78.332122	0.000025	0.000010
	Worst	−1.030079	0.997144	3.000203	0.013295	0.092684	−78.108911	0.038068	0.058219
	Mean	−1.031254	0.998973	3.000081	0.002467	0.033172	−78.261844	0.01329	0.020560
	Rate (%)	90	85	90	70	30	50	40	55
$N = 12$	Best	−1.031627	0.999956	3.000009	0.000031	0.000014	−78.332207	0.000024	0.000007
	Worst	−1.031571	0.999035	3.000126	0.001893	0.073410	−78.138363	0.012648	0.047533
	Mean	−1.031602	0.999318	3.000058	0.000809	0.023561	−78.307064	0.002257	0.011381
	Rate (%)	100	100	95	90	45	70	75	80
$N = 20$	Best	−1.031628	1.0	3.0	0.000023	0.000015	−78.332331	0.000015	0.000008
	Worst	−1.031625	0.999637	3.000085	0.000782	0.042661	−78.327161	0.002134	0.012834
	Mean	−1.031627	0.999750	3.000065	0.000431	0.004982	−78.331905	0.000927	0.008245
	Rate (%)	100	100	100	90	75	90	90	90
$N = 30$	Best	−1.031628	1.0	3.0	0.0	0.000001	−78.332331	0.000004	0.0
	Worst	−1.031627	0.999812	3.000086	0.000240	0.022522	−78.332169	0.000849	0.005968
	Mean	−1.031627	0.999885	3.000041	0.000061	0.003735	−78.332284	0.000593	0.004130
	Rate (%)	100	100	100	100	90	100	95	100

Table 3 Simulation results obtained by MPCOA using different chaotic maps

Map	Stats	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
Logistic	Best	−1.031627	0.999991	3.0	0.000041	0.000071	−78.332322	0.000033	0.000002
	Mean	−1.031625	0.999723	3.000081	0.000575	0.014522	−78.330407	0.001306	0.008225
	Rate (%)	100	100	95	95	70	80	85	90
Tent	Best	−1.031628	1.0	3.0	0.000023	0.000015	−78.332331	0.000015	0.000008
	Mean	−1.031627	0.999750	3.000065	0.000431	0.004982	−78.331905	0.000927	0.008245
	Rate (%)	100	100	100	90	75	90	90	90
Chebyshev	Best	−1.031626	0.999928	3.000009	0.000037	0.000045	−78.332330	0.000061	0.000021
	Mean	−1.031625	0.998522	3.000068	0.001507	0.010334	−78.326544	0.005605	0.007233
	Rate (%)	100	90	95	85	65	85	80	95
Circle	Best	−1.031628	0.999990	3.000008	0.000066	0.000078	−78.332314	0.000025	0.000007
	Mean	−1.031622	0.999135	3.000089	0.000836	0.011372	−78.328211	0.002493	0.008514
	Rate (%)	100	95	95	90	70	85	80	90
Cubic	Best	−1.031628	0.999866	3.0	0.000014	0.000020	−78.332330	0.000066	0.000003
	Mean	−1.031626	0.999671	3.000075	0.000643	0.012365	−78.327665	0.003229	0.007288
	Rate (%)	100	100	100	95	65	80	85	95
Gauss	Best	−1.031628	0.999951	3.000006	0.000050	0.000011	−78.332331	0.000050	0.0
	Mean	−1.031627	0.999102	3.000068	0.001022	0.008550	−78.331439	0.001272	0.007529
	Rate (%)	100	95	100	90	75	90	90	95
ICMIC	Best	−1.031627	0.999988	3.000011	0.000072	0.000066	−78.332314	0.000024	0.000011
	Mean	−1.031623	0.998852	3.000078	0.001458	0.010612	−78.330309	0.002218	0.009137
	Rate (%)	95	95	100	80	60	85	80	85
Sinusodial	Best	−1.031628	1.0	3.0	0.000013	0.000006	−78.332331	0.000009	0.0
	Mean	−1.031627	0.999736	3.000049	0.000145	0.005707	−78.331872	0.001042	0.007147
	Rate (%)	100	100	100	100	80	95	95	100

Table 4 The success rate by different PCOAs

Algorithm	N	f_1 (%)	f_2 (%)	f_3 (%)	f_4 (%)	f_5 (%)	f_6 (%)	f_7 (%)	f_8 (%)
PCOA1	6	80	85	65	55	25	50	40	60
	12	90	85	90	75	40	60	65	75
	20	95	90	95	90	65	80	80	85
	30	100	95	100	90	75	85	90	90
PCOA2	6	80	85	90	60	25	45	45	60
	12	85	90	90	85	45	65	60	70
	20	95	90	95	90	60	80	85	90
	30	100	95	100	95	70	90	90	100
MPCOA	6	90	85	85	70	30	50	40	55
	12	100	100	95	90	45	70	75	80
	20	100	100	100	90	75	90	90	90
	30	100	100	100	100	90	100	95	100

Table 5 Simulation results obtained by different algorithms in 30 runs

Algorithm	Stats	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
PSO	Best	−1.031628	1.0	3.0	0.0	1e−06	−78.332269	3.5e−04	3.2e−05
	Mean	−1.031626	0.999984	3.000029	0.011968	0.004026	−67.541022	1.780975	0.848533
	Std. Dev.	1.0e−06	7.0e−06	1.5e−05	8.6e−03	3.1e−03	7.6e−00	7.7e−01	3.6e−01
CMAES	Best	−1.031628	0.999925	3.0	0.029721	0.0	−78.332301	1e−06	3e−06
	Mean	−1.022527	0.996273	3.000137	0.070160	0.001355	−71.321755	0.275337	0.097554
	Std. Dev.	8.4e−03	1.8e−03	5.8e−05	2.9e−02	8.2e−04	6.2e−00	1.2e−01	6.6e−02
SaDE1	Best	−1.031628	0.999998	3.0	0.000527	2e−06	−78.332295	2e−06	1e−06
	Mean	−1.031627	0.997836	3.0	0.045876	0.003392	−67.814031	0.409426	0.136457
	Std. Dev.	1.0e−06	1.7e−03	0.0e−00	3.2e−02	2.4e−03	8.5e−00	2.3e−01	8.2e−02
SaDE2	Best	−1.031628	1.0	3.0	0.000561	0.0	−78.332306	4.1e−05	1e−06
	Mean	−1.031628	0.999328	3.000032	0.001673	0.008935	−72.136417	0.825630	0.035904
	Std. Dev.	0.0e−00	4.3e−04	1.2e−05	7.3e−04	5.2e−03	5.5e−00	4.8e−01	1.4e−02
MPCOA	Best	−1.031628	1.0	3.0	1.3e−05	6e−06	−78.332331	9e−06	0.0
	Mean	−1.031627	0.999736	3.000049	0.000145	0.005707	−78.331872	0.001042	0.007147
	Std. Dev.	1.0e−06	2.0e−04	3.1e−05	1.1e−04	2.4e−03	1.0e−03	1.5e−04	2.2e−03

these three algorithms are shown in Table 4, where 'PCOA1' and 'PCOA2' represent PCOA+CCIC and PCOA+HSA in Yuan et al. (2007) and (2012), respectively. From Table 4, one knows that with the increase of parallel number N, the success rate of these PCOAs becomes higher. In each case, the MPCOA shows higher success rate than other PCOAs. The simulation results in Table 4 also verify the superiority of MPCOA compared with other PCOAs.

5.5 Comparison of MPCOA with other algorithms

Here, the MPCOA (parameters are: $N = 20$, $P_{\text{cross}} = 0.5$, $P_{\text{merging}} = 0.5$, $\phi = 6$, sinusodial map is used) is also compared with four widely used evolutionary algorithms, they are: particle swarm optimization algorithm (PSO) (Thangaraj et al. 2011), covariance matrix adaptation evolution strategy (CMAES) (Igel et al. 2007), self-adaptive differential evolution algorithm (SaDE1) (Brest et al. 2006) and self-adaptive differential evolution algorithm (SaDE2) (Qin and Suganthan 2005). The simulation results on benchmark functions obtained by these algorithms in 30 runs are reported in Table 5. In addition to the 'Best' and 'Mean', the standard deviation of the mean objective function value (Std. Dev.) is also compared. It can be seen from Table 5 that the proposed MPCOA has achieved the best result among these algorithms in f_4, f_6, f_7 and f_8.

6 Parameter identification of synchronous generator

In this section, simulations are performed to evaluate the performance of the MPCOA for parameter identification of

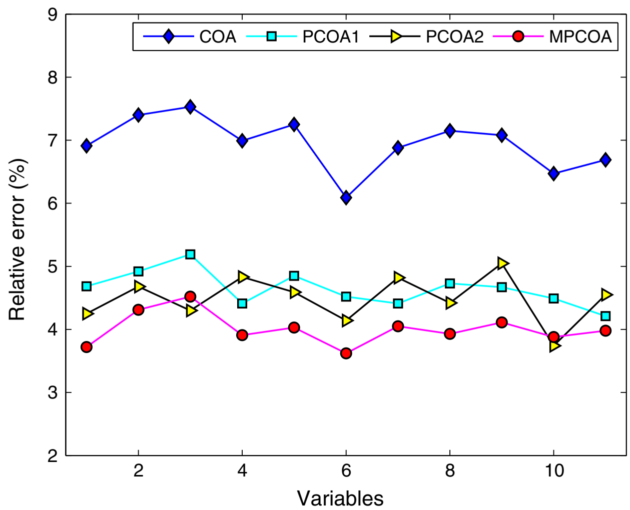


Fig. 7 Average relative errors of 11 variables obtained by different algorithms

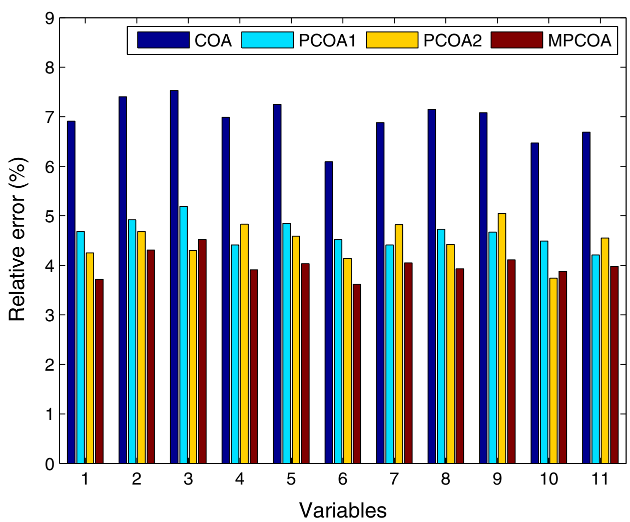


Fig. 8 Average relative errors of 11 variables obtained by different algorithms

synchronous generator. The mathematical model of synchronous generator and the fitness function for this problem are the same in Zhu et al. (2012), Yuan et al. (2012). Similar to Zhu et al. (2012), Yuan et al. (2012), 11 variables values (X_d, $X_d^{'}$, $X_d^{''}$, $T_{d0}^{'}$, $T_{d0}^{''}$, K, X_q, $X_q^{''}$, $T_{q0}^{''}$, M, D) are to be identified by optimization algorithms. The parameters of MPCOA are chosen as: $N = 20$, $P_{\text{cross}} = 0.5$, $P_{\text{merging}} = 0.5$, $\phi = 6$, $l_{\max} = 3,000$, Sinusodial map is used, the number of decision variables $n = 11$. Here, the relative error is used to evaluate parameter identification performance as: $\| \frac{x-\hat{x}}{x} * 100\,\% \|$, where x and $\hat{x}$ are the actual parameter value and the identified value, respectively.

Average relative errors of 11 identified variables by different algorithms repeated for 10 runs are shown in Figs. 7 and 8. The 'COA', 'PCOA1', 'PCOA2' and 'MPCOA' represent original COA, PCOA+CCIC, PCOA+HSA and proposed MPCOA, respectively. It can be seen from Figs. 7 and 8 that the MPCOA has the smallest average relative errors of these 11 variables. The parameter identification results have also verified that the proposed MPCOA has superiority over original PCOAs.

7 Conclusion

In the present paper, a novel MPCOA with cross and merging operation is proposed to improve original PCOA. With the increase of parallel number, MPCOA has better success rate and converge speed. Simulation results show that there is a little difference between different chaotic maps. It is observed that obvious performance improvement is possible by the MPCOA. Benchmark tests and parameter identification results have shown that the MPCOA has better performance than original PCOAs.

References

Alatas B (2010) Chaotic harmony search algorithms. Appl Math Comput 216(9):2687–2699
Arunkumar R, Jothiprakash V (2013) Chaotic evolutionary algorithms for multi-reservoir optimization. Water Resour Manag 27(15):5207–5222
Askarzadeh A (2013) A discrete chaotic harmony search-based simulated annealing algorithm for optimum design of PV/wind hybrid system. Solar Energy 97:93–101
Brest J, Greiner S, Boskovic B, Mernik M, Zumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evol Comput 10(6):646–657
Chen SS (2011) Chaotic simulated annealing by a neural network with a variable delay: design and application. IEEE Trans Neural Netw 22(10):1557–1565
Chen JY, Lin QZ, Ji Z (2011) Chaos-based multi-objective immune algorithm with a fine-grained selection mechanism. Soft Comput 15(7):1273–1288
Cheng MY, Huang KY, Chen HM (2012) K-means particle swarm optimization with embedded chaotic search for solving multidimensional problems. Appl Math Comput 219(6):3091–3099
Coelho LD, Pessoa MW (2011) A tuning strategy for multivariable PI and PID controllers using differential evolution combined with chaotic Zaslavskii map. Expert Syst Appl 38(11):13694–13701
Coelho LD, Mariani VC (2012) Firefly algorithm approach based on chaotic Tinkerbell map applied to multivariable PID controller tuning. Comput Math Appl 64(8):2371–2382
Davendra D, Senkerik R, Zelinka I, Pluhacek M, Bialic-Davendra M (2014) Utilising the chaos-induced discrete self organising migrating algorithm to solve the lot-streaming flowshop scheduling problem with setup time. Soft Comput 18(4):669–681
Hamaizia T, Lozi R, Hamri NE (2012) Fast chaotic optimization algorithm based on locally averaged strategy and multifold chaotic attractor. Appl Math Comput 219(1):188–196
He D, He C, Jiang LG, Zhu HW, Yu GR (2001 Chaotic characteristics of a onedimensional iterative map with infinite collapses. In:

IEEE transactions on circuits and systems I: fundamental theory and applications, vol 48, no 7, pp 900–906

Ho SL, Yang SY (2012) A fast robust optimization methodology based on polynomial chaos and evolutionary algorithm for inverse problems. IEEE Trans Magn 48(2):259–262

Hong WC (2011) Traffic flow forecasting by seasonal SVR with chaotic simulated annealing algorithm. Neurocomputing 74(12-13):2096–2107

Igel C, Hansen N, Roth S (2007) Covariance matrix adaptation for multi-objective optimization. Evol Comput 15(1):1–28

Kromer P, Zelinka I, Snasel V (2014) Behaviour of pseudo-random and chaotic sources of stochasticity in nature-inspired optimization methods. Soft Comput 18(4):619–629

Li B, Jiang WS (1998) Optimizing complex function by chaos search. Cybern Syst 29(4):409–419

Li YY, Wen QY, Zhang BH (2012) Chaotic ant swarm optimization with passive congregation. Nonlinear Dyn 68(1–2):129–136

Liu LZ, Zhang JQ, Xu GX, Liang LS, Huang SF (2013) A modified chaotic ant swarm optimization algorithm. Acta Phys Sinica 62(17):170501

Ma ZS (2012) Chaotic populations in genetic algorithms. Appl Soft Comput 12(8):2409–2424

Okamoto T, Hirata H (2013) Global optimization using a multipoint type quasi-chaotic optimization method. Appl Soft Comput 13(2):1247–1264

Pluhacek M, Senkerik R, Zelinka I (2014) Particle swarm optimization algorithm driven by multichaotic number generator. Soft Comput 18(4):631–639

Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: IEEE CEC 2005. Proceedings of IEEE congress on evolutionary computation, vol 2, pp 1785–1791

Tavazoei MS, Haeri M (2007) Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms. Appl Math Comput 187(2):1076–1085

Thangaraj R, Pant M, Abraham A (2011) Particle swarm optimization: hybridization perspectives and experimental illustrations. Appl Math Comput 217(12):5208–5226

Wan M, Wang C, Li L, Yang Y (2012) Chaotic ant swarm approach for data clustering. Appl Soft Comput 12(8):2387–2393

Wu Q (2011) A self-adaptive embedded chaotic particle swarm optimization for parameters selection of Wv-SVM. Expert Syst Appl 38(1):184–192

Yang DX, Gang L, Cheng GD (2006) Convergence analysis of first order reliability method using chaos theory. Comput Struct 84(8–9):563–571

Yang DX, Li G, Cheng GD (2007) On the efficiency of chaos optimization algorithms for global optimization. Chaos Solitons Fractals 34(4):1366–1375

Yang YM, Wang YN, Yuan XF, Yin F (2012) Hybrid chaos optimization algorithm with artificial emotion. Appl Math Comput 218(11):6585–6611

Yang DX, Liu ZJ, Zhou JL (2014) Chaos optimization algorithms based on chaotic maps with different probability distribution and search speed for global optimization. Commun Nonlinear Sci Numer Simul 19(4):1229–1246

Yuan XF, Wang YN, Wu LH (2007) Parallel chaotic optimization algorithm based on competitive-cooperative inter-communication. Control Decis 22(9):1027–1031

Yuan XF, Wang YN (2008) Parameter selection of support vector machine for function approximation based on chaos optimization. J Syst Eng Electr 19(1):191–197

Yuan XF, Yang YM, Wang H (2012) Improved parallel chaos optimization algorithm. Appl Math Comput 219(8):3590–3599

Yuan XF, Zhao JY, Yang YM, Wang YN (2014) Hybrid parallel chaos optimization algorithm with harmony search algorithm. Appl Soft Comput 17:12–22

Zhu Q, Yuan XF, Wang H (2012) An improved chaos optimization algorithm-based parameter identification of synchronous generator. Electr Eng 94(3):147–153